AMENDMENTS TO THE SPECIFICATION

Please replace paragraph [0019] on pages 8-11 with the following amended paragraph:

At step S310, all clock-domain crossing encountered in a given IC design, are identified. That is, pairs of registers connected through a combinational path, which are clocked by different clocks, are searched for. The clock crossing registers are detected using a synthesized netlist produced by an IC synthesis tool. Synthesis tools produce gate level netlists based, for example, on the register transfer level (RTL) representation. Netlists generally include logical gates such as AND, NAND, NOR, OR, XOR, NXOR, NOT, and the ~~likes~~like. One such synthesis tool is disclosed in a US patent application entitled "An Apparatus and Method for Handling of Multi-Level Circuit Design Data", serial number 10/118,242, assigned to common assignee and is hereby incorporated by reference for all that it contains, especially for its helpful background teaching relating to a synthesis tool. All pairs of crossing registers are saved in a temporary list (hereinafter, the "crossing registers list"). At step S315, it is determined if the crossing registers list is empty. If the list is empty, then execution ends; otherwise, execution continues with step S320. At step S320, a single pair of clock crossing registers is picked from the crossing registers list, namely a clock-domain crossing to be analyzed is selected. At step S330, a search for a structural synchronization cell in the selected clock-domain crossing is performed. A structural synchronization cell may be, but is not necessarily limited to, a double-level register, a recirculation MUX with double-register control, or any other logic that is explicitly used to synchronize the clock-domain crossing. In one embodiment, the user may define such a

synchronization cell and adapt the disclosed method to handle it as a structural synchronization cell. At step S340, a check is ~~preformed~~ performed to determine if a structural synchronization cell was found. If found the execution continues with step S380; otherwise, execution continues with step S350. At step S350, in-depth analysis is ~~preformed~~ performed to determine if the selected clock-domain crossing is stable. The in-depth analysis evaluates a stability function over multiple time frames, where each time frame is defined as a clock cycle driving the relevant registers. The stability function is defined as follows:

$$Ri(t) != Ri(t+1) => Rj(t) = Rj(t+1)$$

where the register pair (Ri, Rj) belongs to the selected clock-domain crossing. Rk(t) will be used to represent the contents of a k^{th} register at time 't'. The stability function implies that R_j must be disabled while R_i loads its new data. This condition ensures the correct stabilization of values in registers across the selected clock-domain crossing. A detailed description of step S350 is provided with reference to Fig. 4. At step S360, a check is ~~preformed~~ performed to determine if step S350 returns an indication of an unstable clock-domain crossing, i.e., if the stability function equals to '0'. If step S360 yields an affirmative answer, then at step S370, the unstable condition, i.e., the time frame that caused the violation as well as the clock crossing registers are added to a report, e.g., a file or a display list; otherwise, the execution continues with step S380. At step S380, a check is performed to determine if the crossing registers' list is empty, namely if all clock-crossing domains, found in the design, were examined. If the crossing registers' list is not empty, then execution continues at step S320; otherwise, execution continues at step S390 where the report is outputted. In one embodiment, the method disclosed may highlight on a

display or other manner the unstable clock-crossing domains in the design. This would allow a user to easily recognize the unsynchronized registers and clocks in the design of ICs.

Please replace paragraph [0022] on page 13, line 11, with the following amended paragraph:

At step S410, the number of time frames is set to the bound number 'K' of time frames over which the analysis to be carried out. In addition, a set of initial states for the analysis process are determined. The initial states may be defined by the user or automatically uploaded. At step S420, a logic circuit which implements the stability function is added to the design. Referring now to Fig. 5 where an exemplary logic circuit 500 implements the stability function, in accordance with present invention, is shown. Circuit 500 includes the following logic gates: exclusive or (XOR) gates 510 and 520, as well as an ~~AND~~ NAND gate 530. Registers R1 and R2 are considered as crossing registers and are part of the original design. Circuit 500 outputs a "stable" signal which determines if the tested clock crossing domain is stable, i.e., if the stability function is valid. If the "stable" signal equals to '0', then the design is considered unstable. Circuit 500 implements the stability function, since the "stable" signal equals to '0' (i.e., unstable) if and only if XOR gates 510 and 520 output '1'. XOR gate 510 outputs '1' only if $R_1(t)$ is not equal to $R_1(t+1)$ (i.e., $R_1(t) != R_1(t+1)$). XOR gate 520 outputs '1' only if $R_2(t)$ is not equal to $R_2(t+1)$ (i.e., $R_2(t) != R_2(t+1)$). In other words, circuit 500 reports instability if both R1 and R2 change data simultaneously which is violating the condition $R_1(t) != R_1(t+1) => R_2(t) = R_2(t+1)$, which is the stability function. Circuit 500 is added to the design on the fly, i.e., the

process of adding circuit 500 to the design is transparent to the user. As method 400 ends, circuit 500 is removed from the design.